



DEPARTMENT OF

Discover. Learn. Empower.

COMPUTER SCIENCE & ENGINEERING

Experiment - 3

Student Name : Anshika Goel

UID: 22BCS10076

Branch: BE-CSE

Section/Group: KRG 2B

Semester: 6th

Date of Performance: 31/01/25

Subject Name: Java

Subject Code: 22CSH-352

1. Aim: Develop a program for

- a) Easy Level: Square Root Calculation
- b) Medium Level: ATM Withdrawal System
- c) Hard Level: University Enrollment System

2. Implementation/Code:

a)

```
import java.util.Scanner;

public class SquareRootCalculator {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number: ");

        try {

            double num = scanner.nextDouble();

            if (num < 0) {

                throw new IllegalArgumentException("Error: Cannot calculate the square
```

```
root of a negative number.");
```

```
}
```

```
System.out.println("Square Root: " + Math.sqrt(num));
```

```
} catch (IllegalArgumentException e) {
```

```
    System.out.println(e.getMessage());
```

```
} catch (Exception e) {
```

```
    System.out.println("Error: Invalid input. Please enter a numeric value.");
```

```
} finally {
```

```
    scanner.close();
```

```
}
```

```
}
```

b)

```
import java.util.Scanner;
```

```
class InvalidPinException extends Exception {
```

```
    public InvalidPinException(String message) {
```

```
        super(message);
```

```
    }
```

```
}
```

```
class InsufficientBalanceException extends Exception {
```

```
    public InsufficientBalanceException(String message) {
```

```
        super(message);
```

```
    }
```

```
}
```

```
public class ATM {
```

```
    private static final int PIN = 1234;
```



DEPARTMENT OF

Discover. Learn. Empower.

COMPUTER SCIENCE & ENGINEERING

```
private static double balance = 3000.0;
```

```
public static void main(String[] args) {
```

```
    Scanner scanner = new Scanner(System.in);
```

```
    try {
```

```
        System.out.print("Enter PIN: ");
```

```
        int enteredPin = scanner.nextInt();
```

```
        if (enteredPin != PIN) {
```

```
            throw new InvalidPinException("Error: Invalid PIN.");
```

```
        }
```

```
        System.out.print("Withdraw Amount: ");
```

```
        double withdrawAmount = scanner.nextDouble();
```

```
        if (withdrawAmount > balance) {
```

```
            throw new InsufficientBalanceException("Error: Insufficient balance. Current  
Balance: " + balance);
```

```
        }
```

```
        balance -= withdrawAmount;
```

```
        System.out.println("Withdrawal successful! Remaining Balance: " + balance);
```

```
    } catch (InvalidPinException | InsufficientBalanceException e) {
```

```
        System.out.println(e.getMessage());
```

```
    } catch (Exception e) {
```

```
        System.out.println("Error: Invalid input.");
```

```
    } finally {
```

```
        System.out.println("Final Balance: " + balance);
```

```
        scanner.close();
```

```
    }
```

```
}
```

c)

```
import java.util.HashMap;
```

```
import java.util.Scanner;
```

```
class CourseFullException extends Exception {  
    public CourseFullException(String message) {  
        super(message);  
    }  
}
```

```
class PrerequisiteNotMetException extends Exception {  
    public PrerequisiteNotMetException(String message) {  
        super(message);  
    }  
}
```

```
public class UniversityEnrollment {  
    private static final int MAX_ENROLLMENT = 2;  
    private static HashMap<String, Integer> courseEnrollments = new HashMap<>();  
    private static HashMap<String, String> prerequisites = new HashMap<>();
```

```
    public static void main(String[] args) {  
        // Defining course prerequisites  
        prerequisites.put("Advanced Java", "Core Java");  
        prerequisites.put("Machine Learning", "Mathematics");
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        try {  
            System.out.print("Enroll in Course: ");  
            String course = scanner.nextLine();
```

```
System.out.print("Prerequisite: ");
String prerequisite = scanner.nextLine();

if (prerequisites.containsKey(course) &&
!prerequisites.get(course).equals(prerequisite)) {
    throw new PrerequisiteNotMetException("Error: PrerequisiteNotMetException
- Complete " + prerequisites.get(course) + " before enrolling in " + course + ".");
}

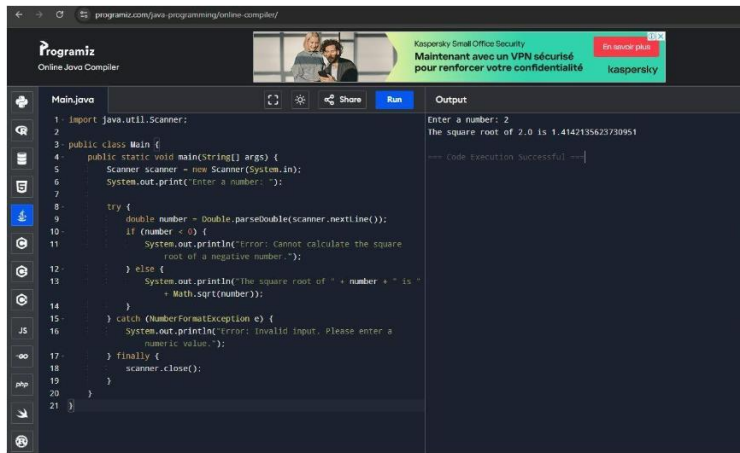
int enrolledCount = courseEnrollments.getDefault(course, 0);
if (enrolledCount >= MAX_ENROLLMENT) {
    throw new CourseFullException("Error: CourseFullException - The course is
full.");
}

courseEnrollments.put(course, enrolledCount + 1);
System.out.println("Enrollment successful for " + course + ".");

} catch (PrerequisiteNotMetException | CourseFullException e) {
    System.out.println(e.getMessage());
} finally {
    scanner.close();
}
}
```

3. Output:

(a)

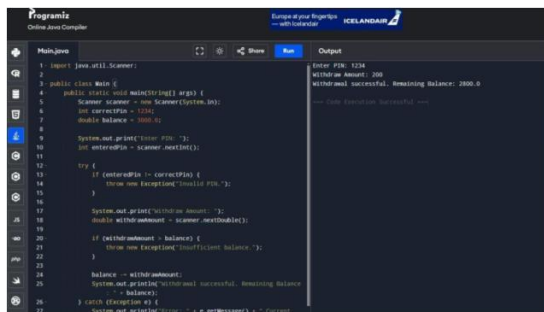


```

1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("Enter a number: ");
7
8         try {
9             double number = Double.parseDouble(scanner.nextLine());
10            if (number < 0) {
11                System.out.println("Error: Cannot calculate the square root of a negative number.");
12            } else {
13                System.out.println("The square root of " + number + " is " + Math.sqrt(number));
14            }
15        } catch (NumberFormatException e) {
16            System.out.println("Error: Invalid input. Please enter a numeric value.");
17        } finally {
18            scanner.close();
19        }
20    }
21 }
  
```

Output: Enter a number: 2
The square root of 2.0 is 1.4142135623730951
Code execution successful

(a)



```

1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         int correctPin = 1234;
7         double balance = 2000.0;
8
9         System.out.print("Enter PIN: ");
10        int enteredPin = scanner.nextInt();
11
12        try {
13            if (enteredPin != correctPin) {
14                throw new Exception("Invalid PIN.");
15            }
16
17            System.out.print("Enter amount: ");
18            double withdrawalAmount = scanner.nextDouble();
19
20            if (withdrawalAmount > balance) {
21                throw new Exception("Insufficient balance.");
22            }
23
24            balance -= withdrawalAmount;
25            System.out.println("Withdrawal successful. Remaining balance: " + balance);
26        } catch (Exception e) {
27            System.out.println("Error: " + e.getMessage());
28        }
29    }
30 }
  
```

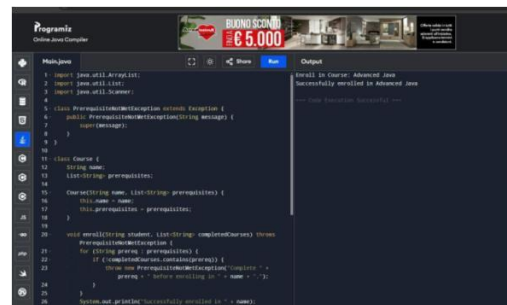
Output: Enter PIN: 1234
Withdrawal amount: 200
Withdrawal successful. Remaining balance: 2000.0
Code execution successful

(b)

6. Learning Outcomes:

- ✓ Exception Handling & Robust Code – Learn to use try-catch, throw, and custom exceptions for handling errors like invalid input, insufficient balance, and unmet prerequisites.
- ✓ User Input & Decision Making – Gain experience in handling user inputs, validating conditions (PIN check, balance check, prerequisites), and controlling program flow.
- ✓ OOP & Data Management – Understand object-oriented principles like custom exception classes and use data structures (e.g., HashMap) for managing enrollments dynamically.

(c)



```

1 import java.util.ArrayList;
2 import java.util.List;
3
4 class PrerequisiteException extends Exception {
5     public PrerequisiteException(String message) {
6         super(message);
7     }
8 }
9
10 class Course {
11     String name;
12     List<String> prerequisites;
13
14     Course(String name, List<String> prerequisites) {
15         this.name = name;
16         this.prerequisites = prerequisites;
17     }
18 }
19
20 void enrollStudent(String student, List<String> completedCourses) throws PrerequisiteException {
21     for (String prereq : prerequisites) {
22         if (!completedCourses.contains(prereq)) {
23             throw new PrerequisiteException("Prerequisite " + prereq + " is not completed before enrolling to " + name + ".");
24         }
25     }
26     System.out.println("Successfully enrolled to " + name);
27 }
  
```

Output: Enter PIN: 1234
Withdrawal amount: 200
Withdrawal successful. Remaining balance: 2000.0
Code execution successful